Ractor Enhancements, 2024 Summer

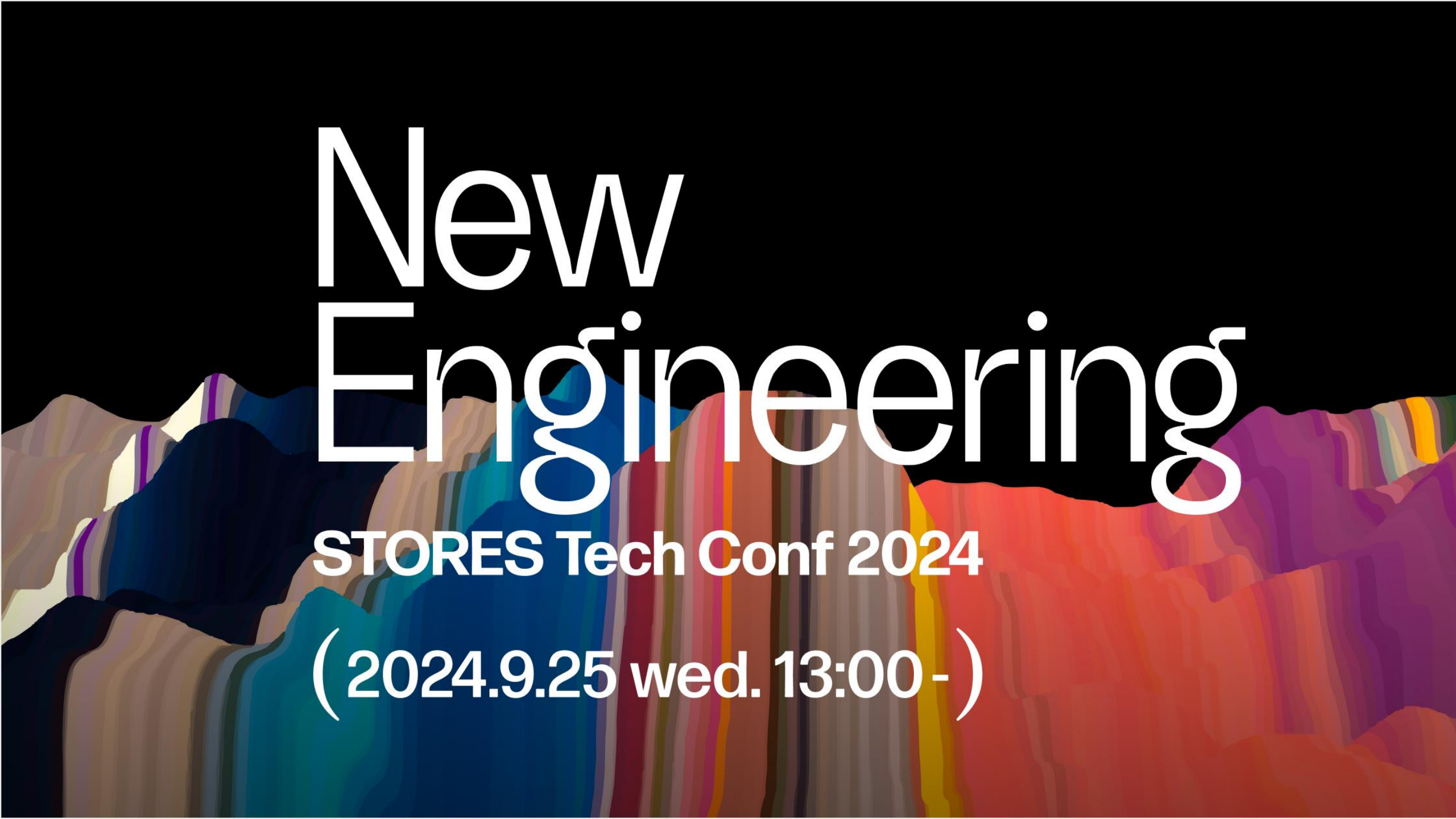
Koichi Sasada
STORES, Inc.

Koichi Sasada

- Ruby interpreter developer employed by **STORES, Inc.** (2023~) with @mametter
  - YARV (Ruby 1.9~)
  - Generational/Incremental GC (Ruby 2.1~)
  - Ractor (Ruby 3.0~)
  - debug.gem (Ruby 3.1~)
  - M:N Thread scheduler (Ruby 3.3~)
  - …
- Ruby Association Director (2012~)

STORES

Glad to be here again this year!


New
Engineering
STORES Tech Conf 2024
(2024.9.25 wed. 13:00 -)

Ractor で目指したい未来
“簡単” に “並列” 並行プログラム in Ruby

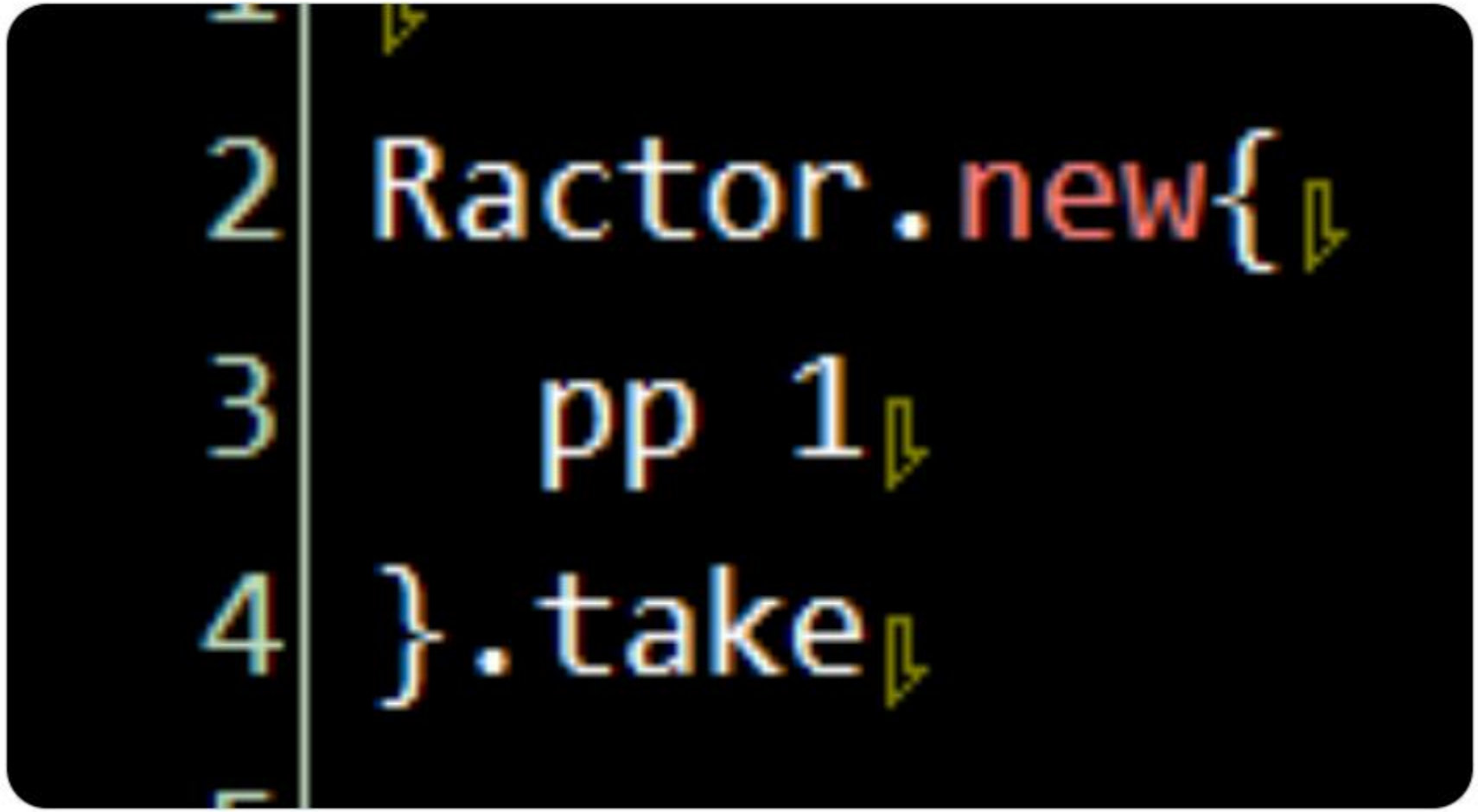
- 短く書ける
- バグらず書ける
- Ruby っぽく書ける（曖昧）
- エコシステムを活用して書ける
- Erlang/Go くらい速いプログラムが書ける
- ビジネスで競争力のあるプログラムが簡単に書ける

RubyKaigi 2024

- Support important features on Ractors
  - **“require”**
  - “timeout”
- Memory management issues on Ractors
- Future enhancement plans
  - GC strategy
  - Proposed APIs

_ko1
@_ko1

このプログラムが色々気にせず動くようになった



午後5:12 · 2024年8月30日 · **1,245** 件の表示

実は今は pp() すらできない

- require ができない
  - 最初の `pp()` 呼び出しは暗黙に `require “pp”`
  - Main Ractor でしか許可していないため

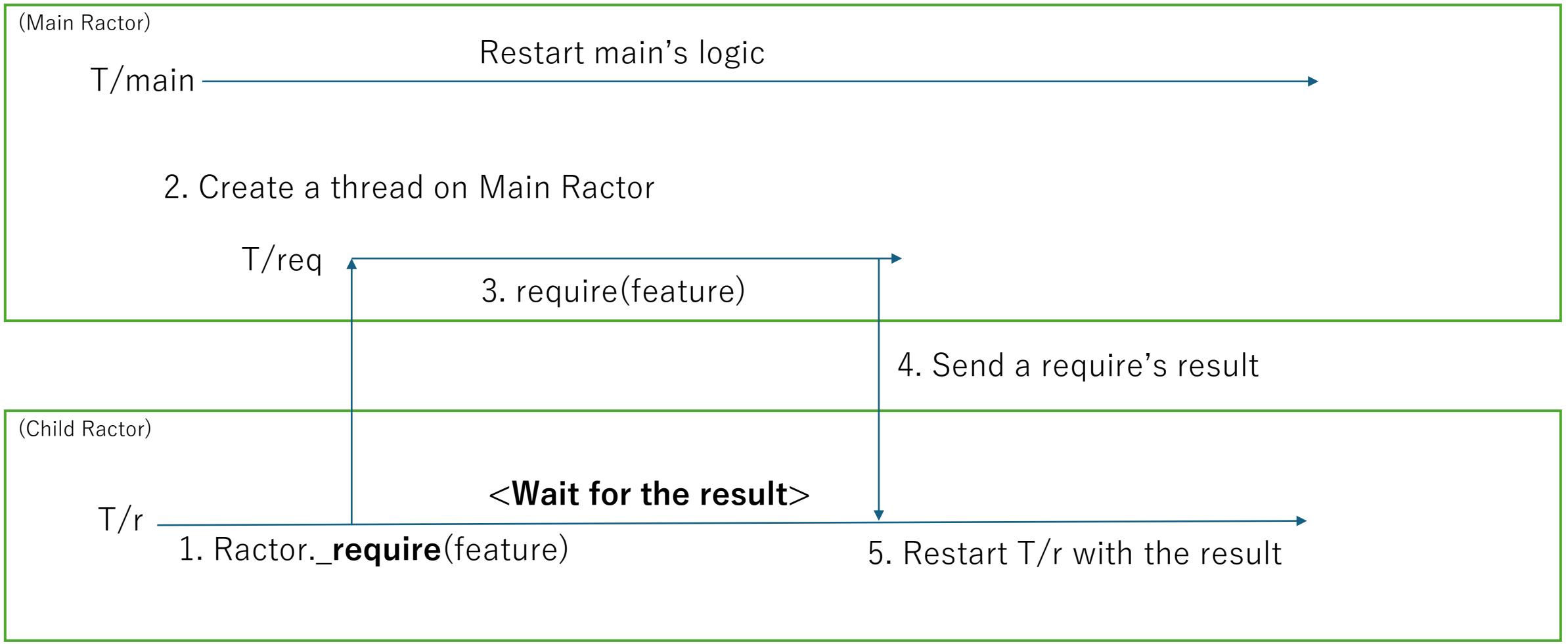
  → 任意の Ractor で任意のプログラムを実行する仕組みを導入

  → 結果を同期的に受け取る仕組みを導入
- `Kernel#require` を上書きする RubyGems が居るので困る
  - `Kernel#require` をいい感じにしても使ってくれない

  → Ractor 起動時に Module を作って Kernel が prepend
- io/console が C-extension だからできない
  - Ractor safe 宣言がいる

  → 付けた

Implement “require” with `Ractor._require(feature)`



Override “require” by Module#prepend only when non-main Ractors are created

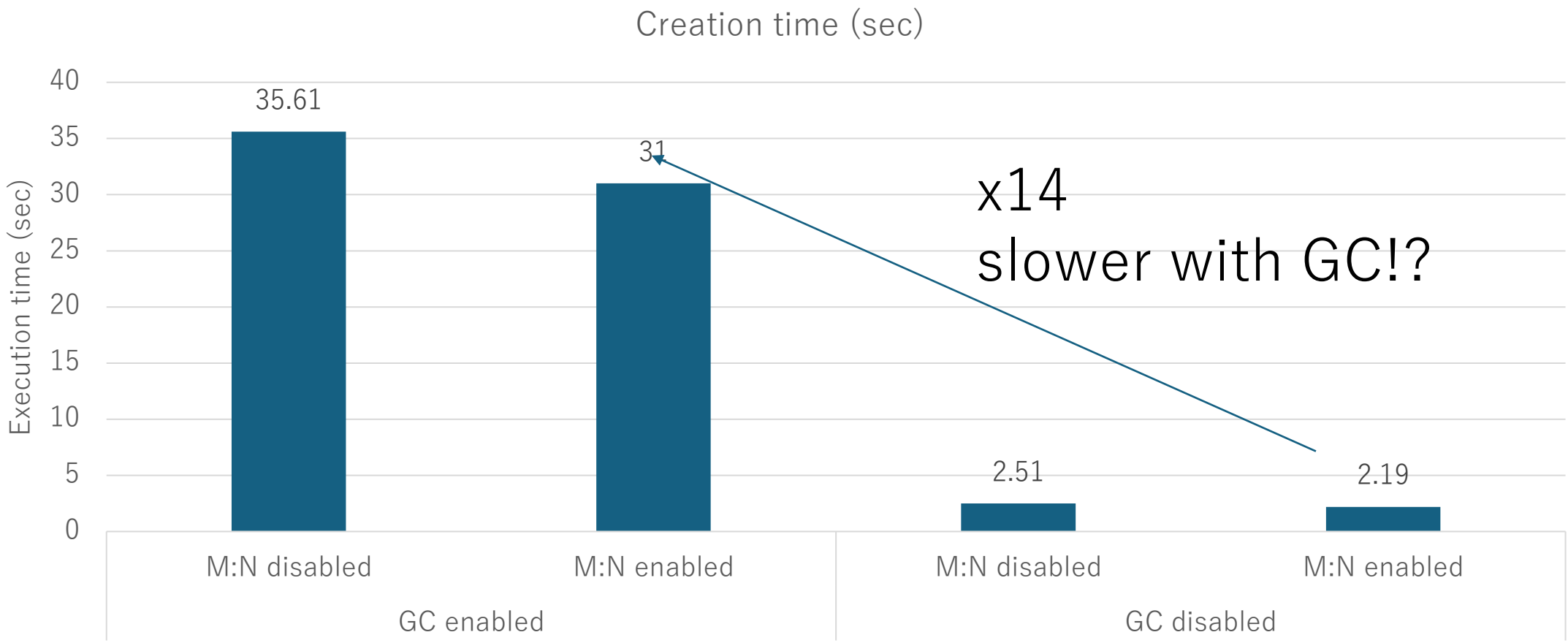
```
Class Ractor
  class << self
    private

    def _require feature
      if main?
        super feature
      else
        Primitive.ractor_require feature
      end
    end

    def _activated
      Kernel.prepend Module.new{|m|
        m.set_temporary_name '<RactorRequire>'
        def require feature
          if Ractor.main?
            super
          else
            Ractor.__send__ :_require, feature
          end
        end
      }
    end
  end
```

他の話

Proposing APIs

- Internal
  - ✅ `Ractor._require(feature)`
  - `Ractor._activated`
- ✅ `Ractor#main?`
- ❔ `Ractor#create_thread`
- Re-implement send/recv/take/yield mechanism
  - `Ractor#yield() with a buffer (sized-queue)`
  - `Ractor::Channel.new`
- `Ractor.[], Ractor.[]= for Ractor local storage`

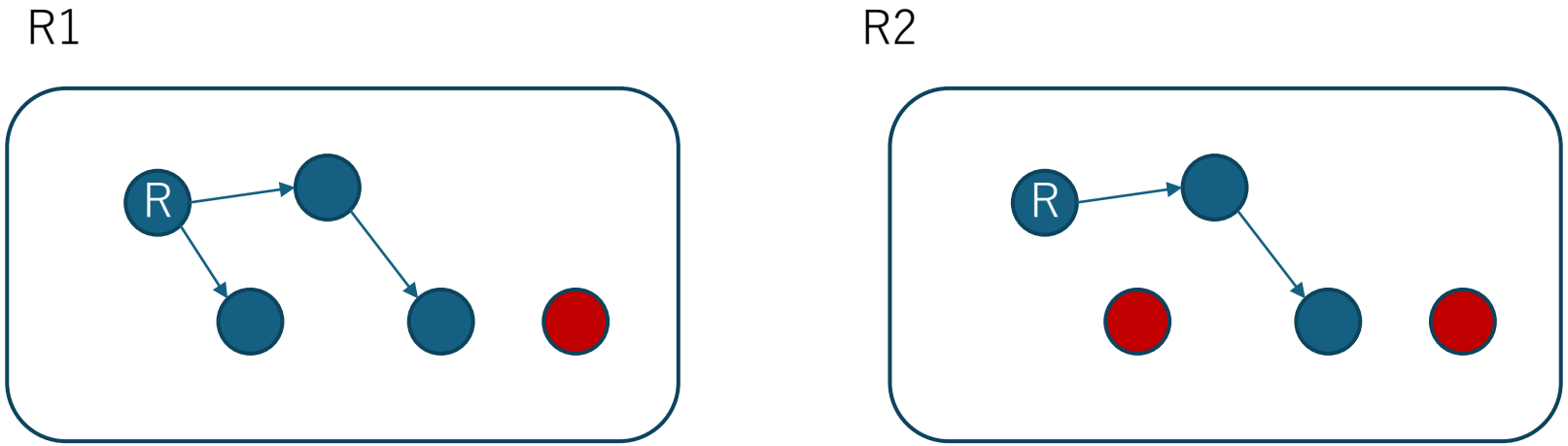
Ring example benchmark results



Quoted from my RubyKaigi 2024 talk

GC tuning

- Ractor aware GC tuning
  - Prepare enough pages for the number of Ractors
- Ractor local GC
  - Need distributed GC techniques
  - Need more memory vs. single heap



Ractor local GC の困難さ

- アルゴリズムの困難さ
  - かっこいいアルゴリズム（分散GC）を採用しようとしたけど難しい
  - 非効率だけど簡単なアルゴリズムに
    - 共有可能オブジェクトを含まない Local GC ＆ 共有可能オブジェクトを含むグローバル GC
    - 分散GCからスレッドローカルGCに縮退
- 実装の困難さ
  - 並列プログラミングしんどくてぇ…
  - 最近 GC のファイルの構造がガラっと変わっちゃってぇ…
    - 全部読み直さないといけない

  → **腰を上げるのがつらい**

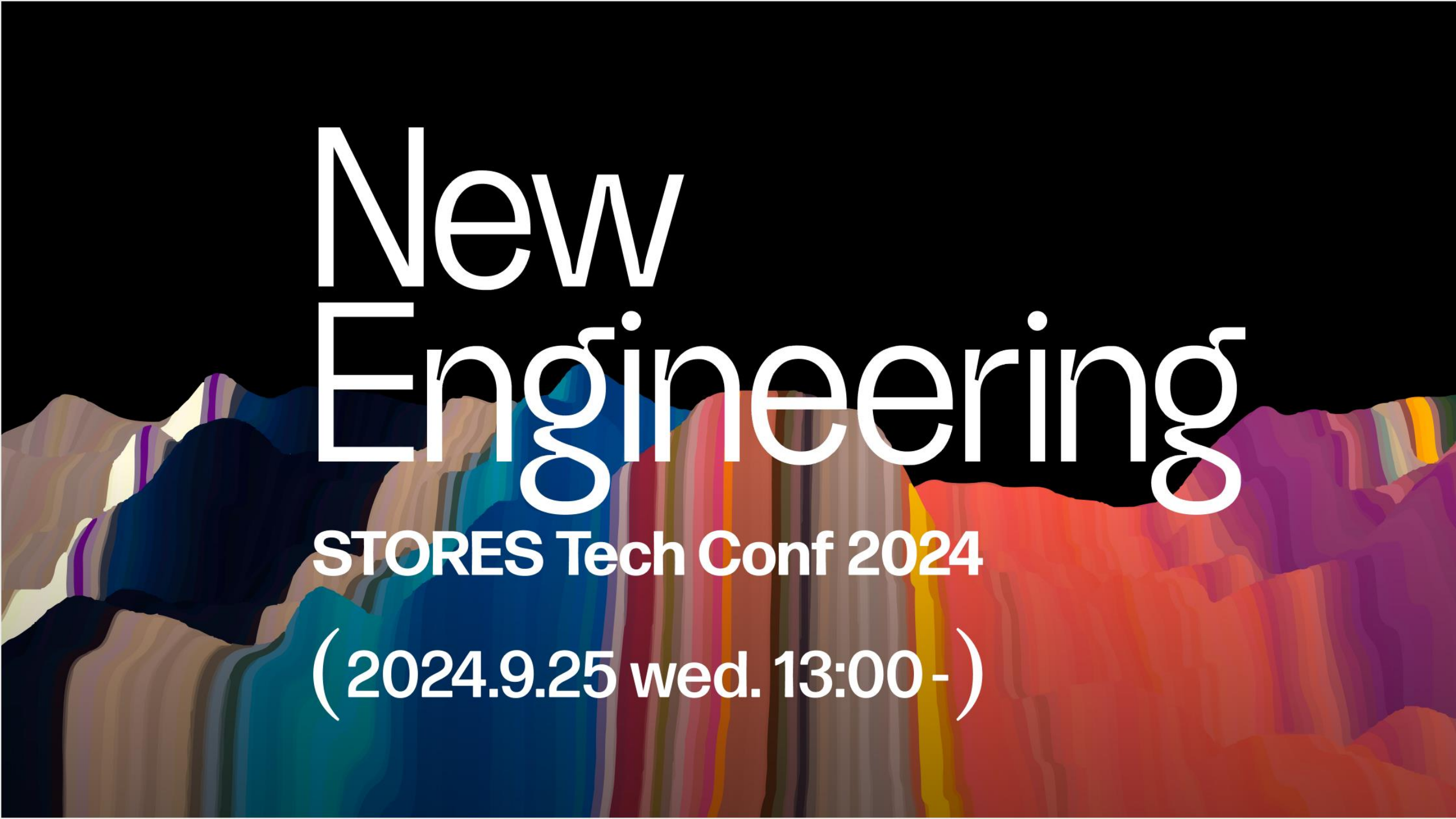
モチベーションの維持

- コードが書けない
  - 歳？
  - Ractorの話をする人がいない

- 興味・関心・要望がある方は議論に付き合って🙏

Summary

- Support important features on Ractors
  - **“require”**
  - “timeout”
- Memory management issues on Ractors
- Future enhancement plans


New
Engineering
STORES Tech Conf 2024
(2024.9.25 wed. 13:00 -)